Personalized Outfit Recommendation System

Kaavish Report
presented to the academic faculty
by

Muhammad Ali	ma02526
Osama Yousuf	oy02945
Syeda Areeba Kazmi	sk02901
Tasneem Adnan	ta02903

In partial fulfillment of the requirements for
Bachelor of Science
Computer Science

Dhanani School of Science and Engineering

Habib University
Spring 2020

Personalized Outfit Recommendation System

This Kaavish project was supervised by:

Dr. Shahid Hussain
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on _____.

Dedication

For ammi, abbu, and pappu.

Acknowledgements

We want to thank the CS faculty and ...

Abstract

Abstract goes here

Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement

Domain: Fashion e-commerce.

Facts and Figures:

1. Personalized shopping is the future of commerce. It is reported that on average today, at least 27% of retail site revenue in fashion, which totals to around $870 million, comes from personalized recommendations systems. [1]

2. In 2013, over 85% of Amazon sales revenue came through personalized recommendations. [2]

3. Despite all its potential, the Pakistani fashion industry is lagging behind in keeping up with such advances in personalized recommendation systems. [3]

Statement: One of the biggest problems in fashion retail is product curation. Retailers have to spend a large amount of time to come up with different combinations of their products that would as a whole, go well as an outfit, and even then, the options aren't really personalized. A customer buys a new shirt, brings it home, and hangs it up, only to find that the shirt stays in their closet for weeks because they're not sure what to pair it with. This also means a loss in conversion rates and potential revenue at the side of the retailer.

1.2 Proposed Solution

As already described, fashion retailers spend a lot of time manually curating their products, and according to a report published by emerj.com (database of reports on AI technology), at least 40% of potential revenues are lost because of poor outfit

recommendations. We see a business opportunity in this problem, and so the idea behind the project is to solve it by addressing the key issue, product curation, by providing expert recommendations across different clothing items to the end-consumer at the point of sale or as a standalone service.

Our solution is a web application that would allow shoppers to visually search the catalogue of e-commerce stores by uploading pictures of outfits they like or taking a photo with their phone's camera. Using Computer Vision, the outfit would be broken down into its constituent parts (eg. shirt, pants, belt, sneakers) and identical and/or visually similar items from the store would be shown at the same place. This would allow shoppers to quickly and conveniently shop for items they see on social media, significantly increasing conversion rate.

1.3 Intended User

According to a recent study, millennials and Generation Z are the most coveted demographics for e-commerce stores. They do 60% of their shopping online [4] and make more apparel purchases than other generations [5]. On average, they spend three hours per day on their phones, mostly on social media platforms such as Facebook and Instagram, constantly consuming and interacting with visual content.

Our intended user are these audiences, and in order to appeal to them, it is essential for e-commerce stores to change the way shoppers interact with their stores. When someone sees their favourite Instagram influencer wearing an outfit that they want, searching for each piece of that outfit via text is not only cumbersome, it is inefficient and unlikely to yield accurate results. In order to allow customers to shop the same way they interact with social media i.e. via images, fashion e-commerce stores are increasingly looking to Artificial Intelligence and Computer Vision powered solutions.

To ensure practicality and applicability, we have been gathering and incorporating feedback from HU faculty as well as industry professionals from Love For Data, Daraz.pk, and PCSIR.

Our application would primarily provide two sets of recommendations when an item is being viewed by a user:

1. Items **visually similar** (and of the same type eg. shirt for shirt) to that currently being viewed, increasing the likelihood that shoppers will find an item they like that is available in their size and at an agreeable price point.

2. Items **visually complementary** to that being viewed, allowing users to "Complete the Look". This allows stores to upsell and increase Average Order Value

(AOV).

In addition, we will also actively look into personalized fashion recommendations based on user purchase history and general trends.

1.4 Key Challenges

A few key challenges that we have identified to foresee in this project are listed below, along with possible ways to address them.

1. We require a dedicated machine in one of the University's labs for hosting our web-server and preferably also a web hosting service. A possible remedy is to take use of local hosting. However, it must be noted that this would increase difficulty in collaborating.

2. Similarly, unavailability of a GPU can hinder the precision of the recommendation system, which would be created entirely from scratch. A simple remedy is to resort to cloud-based services for GPUs such as AWS or Google CoLab.

3. Another challenge would be to clean and curate the dataset as per our requirements and domain. Pre-existing datasets (explained further in chapter 3) may not be exactly in a usable condition out-of-the-box. Therefore, the data would then need to be scraped and cleaned manually which can be cumbersome. A remedy would be to maintain a clean storage format from the get-go.

4. In addition to this, lack of relevant technical knowledge on part of the team is also a challenge. This will be addressed by taking tutorials and online courses.

5. At the same time, insufficient knowledge and expertise in the domain of e-commerce requires us to reach out to industrial partners and professionals from Daraz and Telemart, whose unavailability at times can obstruct the smooth progression of our project.

2. Literature Review

Recommendation system is one of the most important tools for most websites. Various websites, recommend a variety of products to their users, for e.g. books, movies, songs, ads, etc. But these products can't be separated, while there are products which can be separated into different parts and categories, like outfits. As online shopping and fashion-focused social networks are growing rapidly, there is a great need for intelligent fashion recommendation. Retailers have to spend a large amount of time to come up with different combinations of their products that would as a whole, go well as an outfit, and even then, the options aren't really personalized.

Hence, our project will have two main components, a web application and a recommendation system. Therefore, our literature review is divided into two parts.

2.1 Web-Application

The literature review for our web-application is divided into three domains. Technologies used for client-side, server-side and the API for establishing the interaction between them.

Table 2.1, Table 2.2, and Table 2.3 offer a detailed comparison between different communication interfaces.

Server-side Technologies			
Django	**Tornado**	**ASP.NET**	**Node.JS**
In Python, good support for ML out of the box.	Also In Python, good support for ML out of the box.	In C#, Visual Basic, F#, high dependency on external ecosystems for ML, not as intuitive.	In JavaScript, moderate support for ML out of the box.
High community support.	Low community support.	Moderate community support.	High community support.
Not completely asynchronous, though offers support.	Highly synchronous.	Supports both, synchronous as well as asynchronous operations.	Supports both, synchronous as well as asynchronous operations.
Supports direct integration with React/Front-end frameworks/AJAX	Third-party libraries for integration.	Completely different eco-system	Direct integration with React and Angular.
Offers abstraction, is high level.	Also high level.	Relatively low-level.	Relatively low-level.

Table 2.1: Technology Comparison: Server-side Communication

Client-side Technologies		
Django	**React**	**Angular**
Bound to a Django server, imposes design and flow restrictions.	Not bound to a specific server, gives high design freedom.	Not bound to a specific server, gives moderate design freedom.
Complete framework, potential overheads.	Only a library on top of JS, low overheads.	Complete framework, potential overheads.
Low community support.	High community support.	Moderate community support.
Offers high throughput in development.	Offers high throughput performance-wise.	Offers high throughput performance-wise.

Table 2.2: Technology Comparison: Client-side Communication

API Technologies	
REST	**GraphQL**
Every resource is identified by a unique URL having its own end-point that requires router handlers.	Resources are identified by fields in their schema representation with individual resolvers.
Each request calls exactly one route handler, for a nested query, this leads to overheads.	Each query can call many resolvers to construct a nested response with multiple resources.
The response shape has to be manually constructed and is kept fixed.	The response shape is constructed automatically to match the query shape.
As a result, resource type, shape, & fetch query are coupled.	Everything is de-coupled leading to higher scalability.

Table 2.3: Technology Comparison: API-level Communication

Our Approach

Based on the project requirements and keeping the communication channels in mind, the chosen technology stack for the client-end is: TypeScript, React, and React Apollo (for interacting with the GraphQL API), and for the server-end is: Python, Django, and Graphene Django (for creating the GraphQL API).

2.2 Recommendation System

This literature review regarding recommendation system, aims to compare the various methods and techniques studied in the existing work for recommendation systems. It is divided into two domains, the dataset and recommendation engine.

2.2.1 Dataset

The largest and best annotated publicly available dataset in the domain of fashion is **DeepFashion**.

The DeepFashion dataset consists of more than 800,000 richly annotated images, ranging from well-posed shop images to unconstrained consumer photos, making it twice the size of the largest previously available dataset. Each image in the dataset is labeled with 50 categories, 1,000 descriptive attributes, and clothing landmarks.

Another strength of the DeepFashion dataset is that it contains rigorous benchmark for testing the performance of algorithms for clothes recognition. The three benchmarks it contains are clothing attribute prediction, in-shop clothes retrieval,

and cross-domain clothes retrieval, a.k.a. street-to-shop. All three are relevant for the scope

1. **Category and Attribute Annotation:** Each image is labelled with a single category label. There are a total of 50 mutually exclusive categories, labelled by human annotators. The dataset also contains 1000 attributes which are extracted from image metadata. However, these are not utilized by our model. There are 63,720 diverse images in this benchmark.

2. **In-Shop Clothes Retrieval:** This task determines if two in-shop images belong to the same clothing item. This benchmark contains 54,632 images of 11,735 clothing items.

3. **Consumer-to-Shop Clothes Retrieval:** Aimed at matching consumer-taken photos with their shop counterparts. Contains 251,361 consumer-to-shop image pairs.

	DCSA [3]	ACWS [1]	WTBI [12]	DDAN [4]	DARN [10]	**DeepFashion**
# images	1856	145,718	78,958	341,021	182,780	**>800,000**
# categories + attributes	26	15	11	67	179	**1,050**
# exact pairs	N/A	N/A	39,479	N/A	91,390	**>300,000**
localization	N/A	N/A	bbox	N/A	N/A	**4∼8 landmarks**
public availability	yes	yes	no	no	no	**yes**

Table 1. Comparing DeepFashion with other existing datasets. DeepFashion offers the largest number of images and annotations.

Figure 2.1: Dataset Comparison

Our Approach

We will be using DeepFashion dataset as it contains large number of outfit images which are vastly categorized and labelled to train our model and benchmark performance. We will then train the model further on our local dataset, scraped from local stores like, Export Leftovers, J. Furor, Zellbury, etc. As eastern data is comparatively less, our model would suffer from overfitting if used as the only source.

2.2.2 Recommendation Engine

Viet et al. [6] learned featured transformation for measuring compatibility between different pairs of items using a Siamese CNN architecture, whereas, McAuley et al. [7] used parametric distance transformation for the same purpose. Parametric distance transformation assigns the lowest distance to pairs of clothing which fit well.

However, these techniques only gave the matching pairs of clothing and didn't take the personalization issue into account. The initial attempt to explore the personalized outfit recommendation was made by Hu et al. [8], using functional tensor factorization method, however, they used hand-crafted features.

Recently, researchers have explored deep networks and have begun to apply deep learning to recommendation systems. Table 2.4, Table 2.5, and Table 2.6 gives a detailed comparison between different approaches used in multiple research papers.

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
Large Scale Visual Recommendations From Street Fashion Images.	Recommends visually complementary items using Deterministic Fashion Recommenders (DFR) and Stochastic Fashion Recommender.	Fashion-136K, Fashion-350K, Fashion-Q1K datasets.	Not mentioned.	Not mentioned.

Table 2.4: Technology Comparison: Recommendation Engine

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
Learning visual similarity for product design with convolutional neural networks.	Training with stochastic gradient descent. t-SNE algorithm to visualize the result.	Dataset made from Houzz.com.	MTurk to collect the necessary bounding boxes. AlexNet to detect both near and exact duplicates.	Using a Grid K520 GPU it took 100 ms to compute.
MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching.	Deep convolutional network extracts features and a network of three FC layers computes similarity. Feature network is influenced by AlexNet. ReLU for the convolution layers.	UBC patch dataset.	AlexNet	18 hours to 1 week to train the full network.
FashionNet: Personalized Outfit Recommendation with Deep Neural Network.	VGGNet feature network for feature extraction and multilayer fully connected network for computing clothes compatibility.	Dataset collected from Polyvore	VGGNet. Caffe.	Not mentioned.

Table 2.5: Technology Comparison: Recommendation Engine

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations.	FashionNet simultaneously predicts landmarks and attributes. Network structure similar to VGG-16 except last convolutional layer, which is replaced by three branches of layers. Regression loss for landmark localization. Softmax loss for the predictions of categories. Triplet loss for metric learning	800, 000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos. Mogujie, Forever21 and Google Images	FashionNet to demonstrate usefulness.	Not mentioned.
Using Very Deep Autoencoders for Content-Based Image Retrieval	Train RBM by standard contrastive divergence learning procedure. To reduce noise, we use the probabilities rather than the stochastic binary states. Fine Tune auto-encoder by back propogation.	Preprocessed 1.6 million 32×32 color images. CIFAR-10 dataset.	Restricted Boltzmann Machines.	2 days on Nvidia GTX 285 GPU

Table 2.6: Technology Comparison: Recommendation Engine

Our Approach

The paper "MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching" [9], published by Google Research in 2015 served as the inspiration for the FashionNet architecture proposed by the creators of DeepFashion. They use a single network to simultaneously predict both landmarks and attributes. The structure of this network is similar to VGG-16, with the last convolutional layer replaced by three

branches of layers carefully designed for clothes.

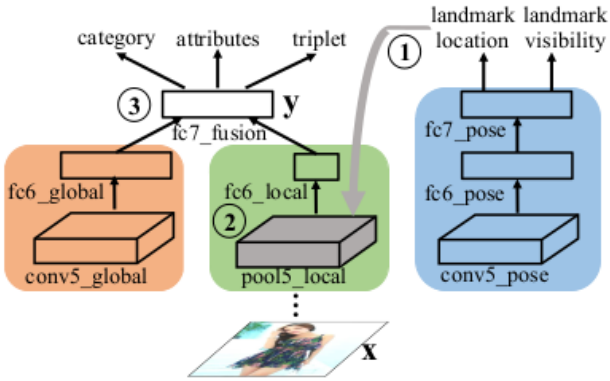

Figure 5. Pipeline of FashionNet, which consists of global appearance branch (in orange), local appearance branch (in green) and pose branch (in blue). Shared convolution layers are omitted for clarity.

Figure 2.2: FashionNet Architecture [10]

However, since the only task we wish to perform in this project is in-shop retrieval and in the interest of making our model simpler, we decided to use an approach relying on only a feature network and use Euclidean distance to find similar vectors. Our feature network will consist of a ResNet50 model trained on the DeepFashion clothing classification benchmark using transfer learning. This feature network will convert images into a 512-dim feature vector. We will then query the database to retrieve images who's feature vector has minimum euclidean distance from the query feature vector. In initial testing thus far, the approach works well. We will continue updating the literature review as we experiment and make our model more accurate and robust.

3. Software Requirement Specification (SRS)

This chapter provides detailed specifications of the system under development.

3.1 Functional Requirements

This section describes each function/feature provided by our system. These functions are logically grouped into modules based on their purpose/users/mode of operations etc (as per our system):

3.1.1 Web Application

- Allows customers to upload a photo of an outfit.
- Displays constituent items of uploaded outfit.
- Allows customers to click on a constituent item and view items identical/similar to it.
- Allows customers to click on an item and open it's product page.
- Displays item title, description, price, photos and sizes on product page.
- On the product page, displays recommendations for similar products, as received from the recommendation engine.
- On the product page, displays recommendations for complementary products, as received from the recommendation engine.
- Allows customer to add item to cart.

- Allows new customers to sign-up.
- Allows returning customers to login.
- Saves user history to database.
- For existing customers with a user history: Displays recommendations based on user history, as received from the recommendation engine, on the home page.
- For existing customers with a user history: Displays recommendations based on user history, as received from the recommendation engine, on product pages.
- For new customers or existing customers without user history: Displays items currently trending on the store, on the home page.
- Admin has exclusive access to an admin panel that allows them to add, remove and modify items on the website.
- Admin can view customer activity.

3.1.2 Recommendation Engine

The engine can be roughly compartmentalized as follows:

- On uploaded photo, calls feature extraction model and performs multiple object detection. To bounding boxes of each individual piece of clothing in the outfit.
- Crops each bounding box in image and returns them. Each bounding box represents an article of clothing.
- Upon receiving which bounding box has been selected by the user, it passes that to nearest neighbour model. Returns ids of n closest neighbours of that item from the database and ids of n best complimentary items.

3.1.3 Database

- Should store id, title, price, description, sizes, category and tags for each item.
- Should store user and admin account information.
- Should store user profile information, including past logs, search history, purchase history, as well as other related actions.

3.1.4 Web Scraper

- Crawl local fashion stores 'Furor', 'Export Leftovers', 'J.', 'Zellbury', etc.
- For each item of men's clothing on the store, save the photo, description and price locally.
- Add timed cron job functionality so that the database can be verified and kept up to date without discrepancies.

3.2 Non-functional Requirements

3.2.1 Performance Requirements

- High performance of the computer on which the server is hosted is needed to cater to thousands of users.
- Fetching the dashboard to view information and recommended outfits shall take no longer than 5 seconds.

3.2.2 Safety Requirements

- The system must not halt or lag, especially during the update time and must not go down under high traffic. In order to ensure safety of the server, it is suggested that it is hosted on two computers - one kept as a backup.

3.2.3 Security Requirements

- It must be ensured that only the authorized admins, with valid user credentials, have access to the data of the users in order to ensure user privacy.
- The system will use databases from authentic sources and fashion stores.

3.2.4 User Interface

- The UI/UX flow needs to be intuitive as well as modern.

3.2.5 Error Handling

- The system prevents data loss by carefully handling all expected and non-expected errors.

3.3 External Interfaces

3.3.1 User Interfaces

Customer Interface

- Homepage

  This interface would be visible to all the users and would lead to multiple other interfaces such as Login, Clothing categories, User profile, etc. This is shown in Figure 3.1.

- Registration

  This page allows a new user to create an account by filling the mandatory fields of 'User Name', 'Email', 'Password' and 'Confirmed Password'. In case of valid details, user will be able to login, else they are redirected to the same sign-up page. This is shown in Figure 3.2.

- Login

  This interface enables user to log into the system using valid credentials, and redirects him/her to the homepage if the credentials are validated, otherwise an error message is displayed. Login interface requires 'User Name' and 'Password' as the mandatory fields. This is shown in Figure 3.3.

- Profile

  A user would be able to see this interface if they have created an account and are logged into the system. This interface would enable them to view their account details and their previously searched/recommended outfit statistics. This is shown in Figure 3.4.

- Product Display

  A user would be able to see different product categories along with the details of each product available in the stock. This is shown in Figure 3.5, Figure 3.6, and Figure 3.7.

- Upload Image

  A user would be able to upload image of his/her clothing item, in order to search visually similar or complementary items. This is shown in Figure 3.8 and Figure 3.9.

- Cart

  This interface enables a user to view their selected recommended outfits and would show them the third-party referral links to each of their chosen outfits. This is shown in Figure 3.10.

- Our Brands

  This interface would display the vendors and the third-party brands with whom we will partner up with. A user would be able to retrieve products based on any specific store from these brands. This is shown in Figure 3.11.

System Admin Interface

- Login

  Using the login interface, the system admin would log into the system using his/her valid credentials and would be redirected to the system admin homepage. This is shown in Figure 3.3.

- Homepage

  This interface enables the system admin to get redirected to multiple other navigation pages such as User Details, and Vendor details. This is shown in Figure 3.1.

- User Details

  This interface contains user details such as their personal information, their cart details and their preferred trend statistics. This is shown in Figure 3.12 and Figure 3.13.

- Vendor Details

  This interface allows the system admin to view details of the vendor such as the trends of their most recommended outfits, new additions to their outfit database, top users visiting the respective vendor page (using the referral link). This is shown in Figure 3.14 and Figure 3.15.

GUI Mockups

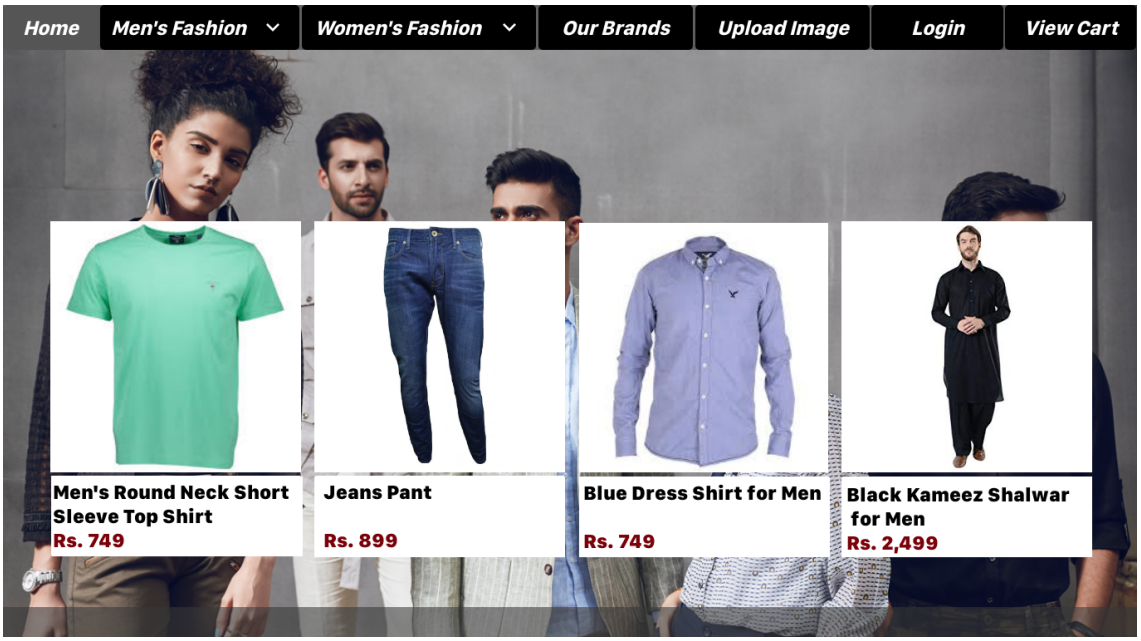

Figure 3.1: Home Screen

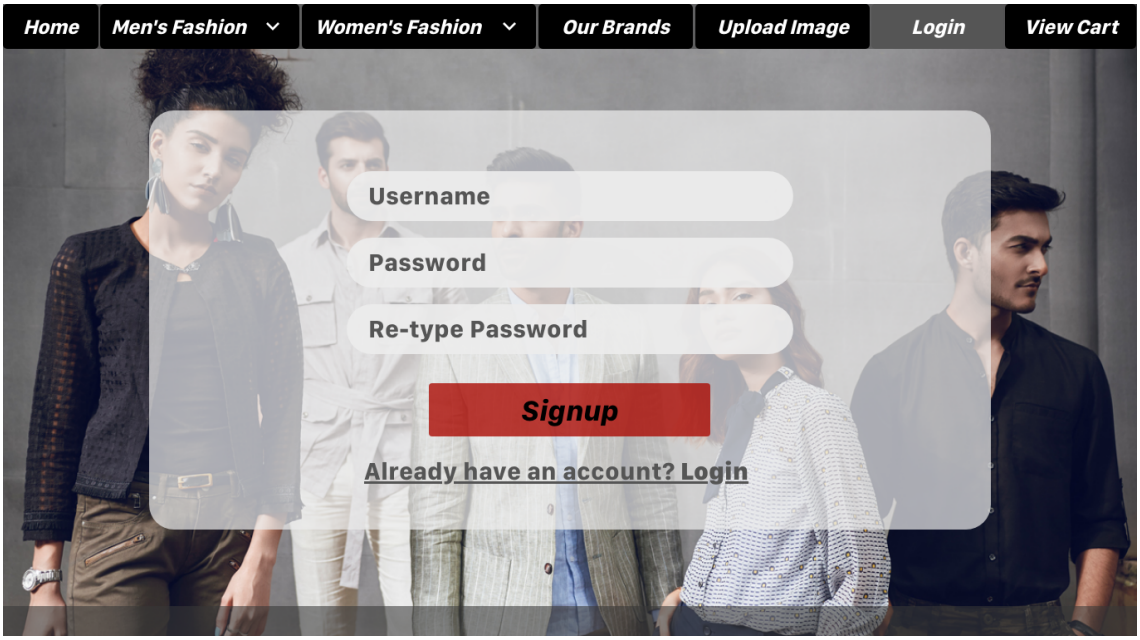

Figure 3.2: Sign-up Screen

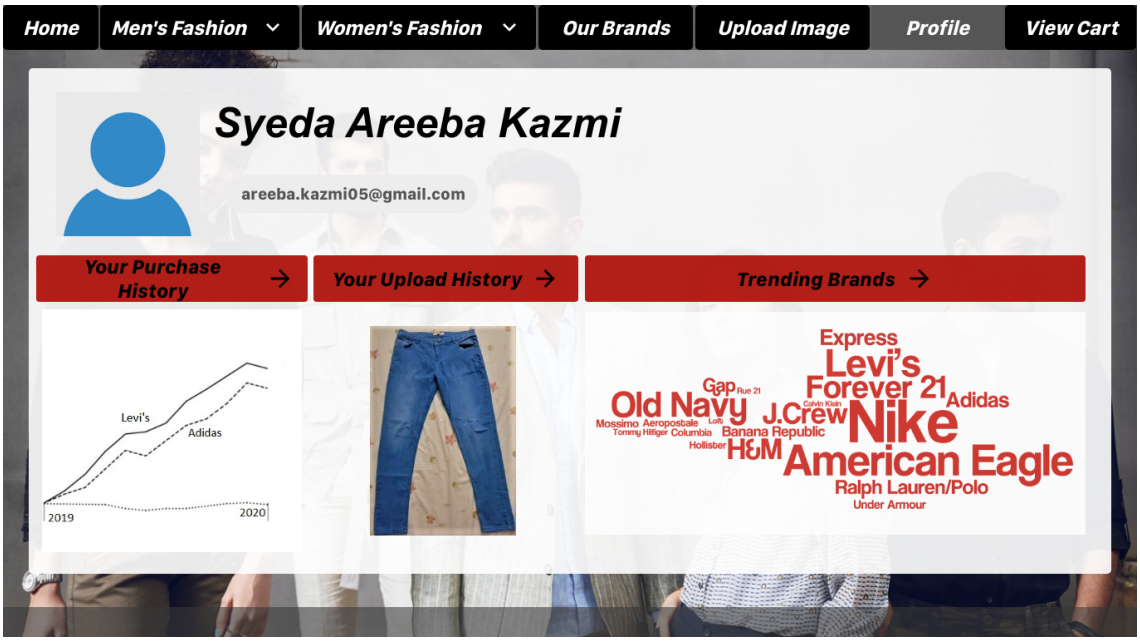

Figure 3.4: User Profile Screen

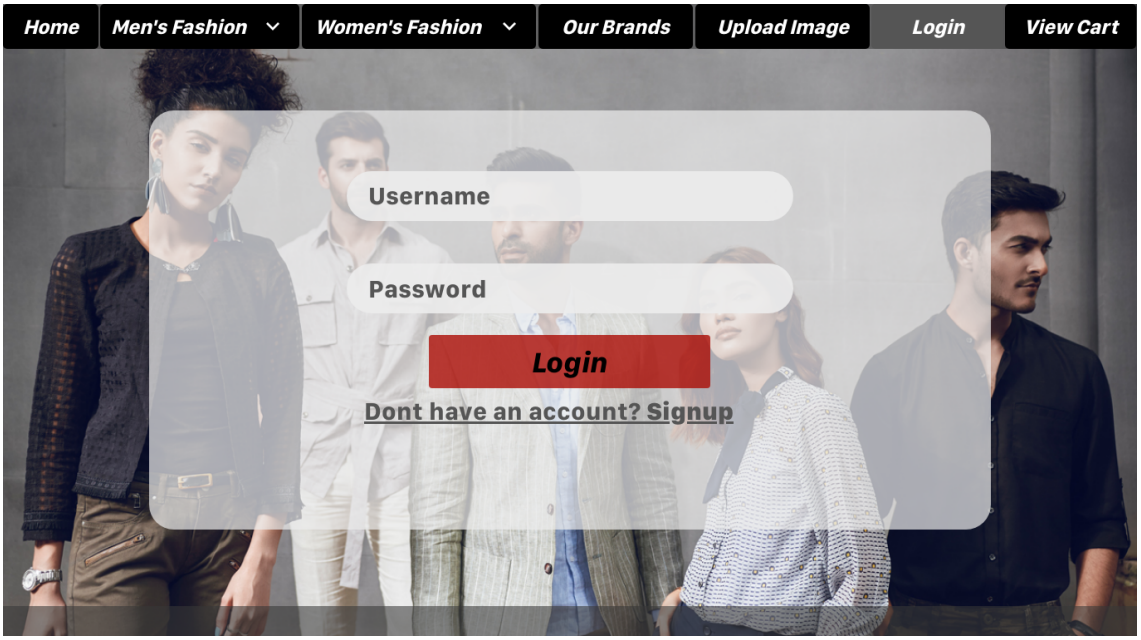

Figure 3.3: Login Screen

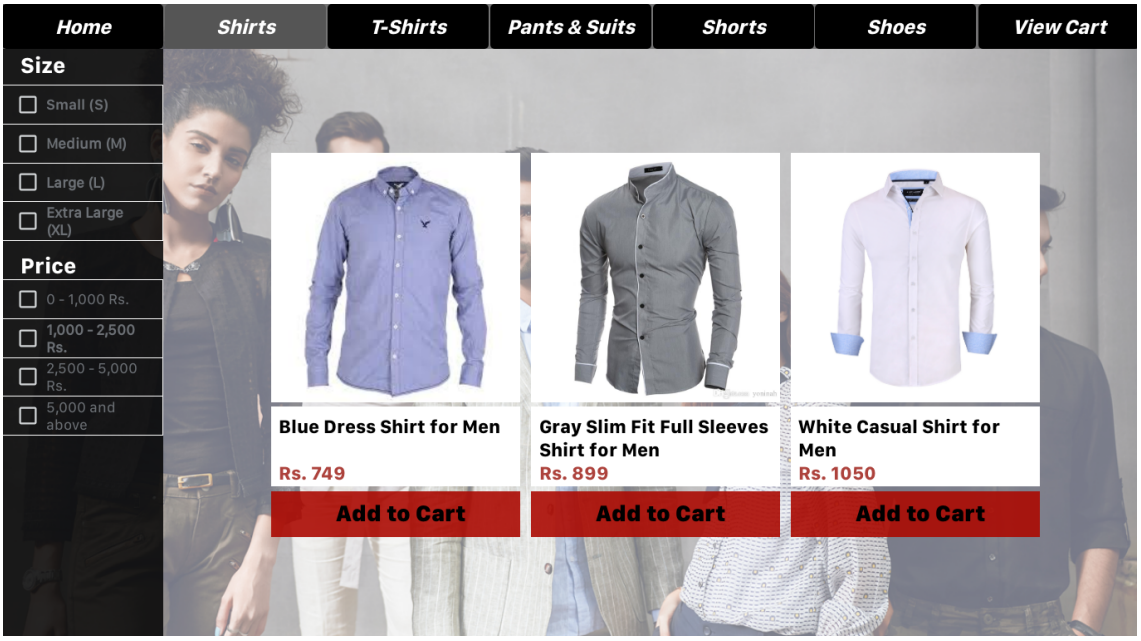

Figure 3.5: Shirt Category Screen

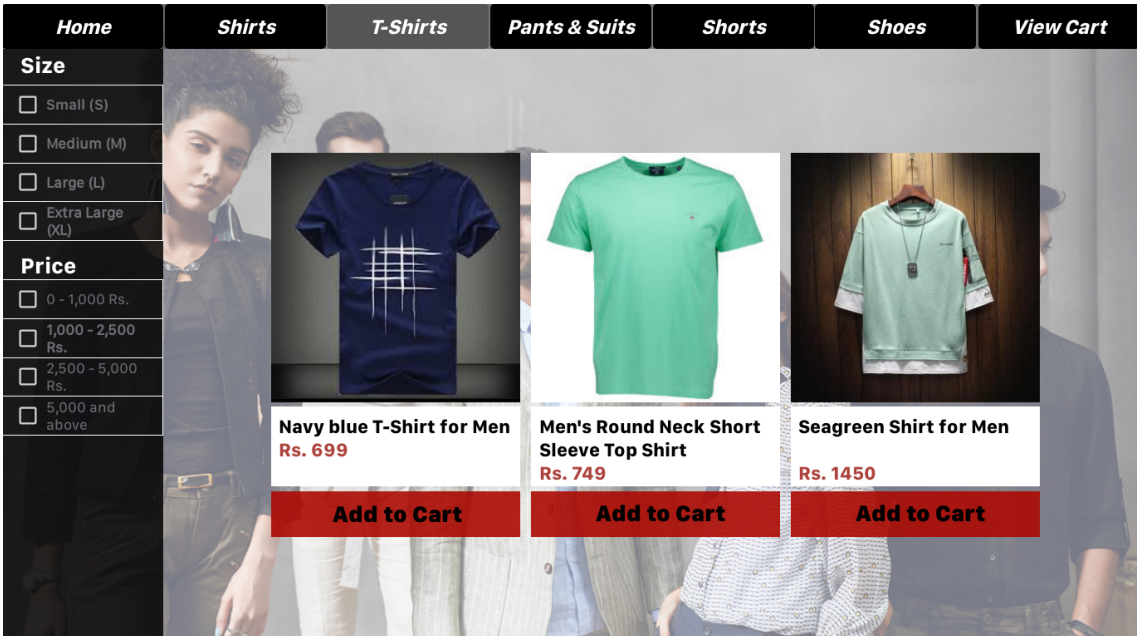

Figure 3.6: T-Shirt Category Screen

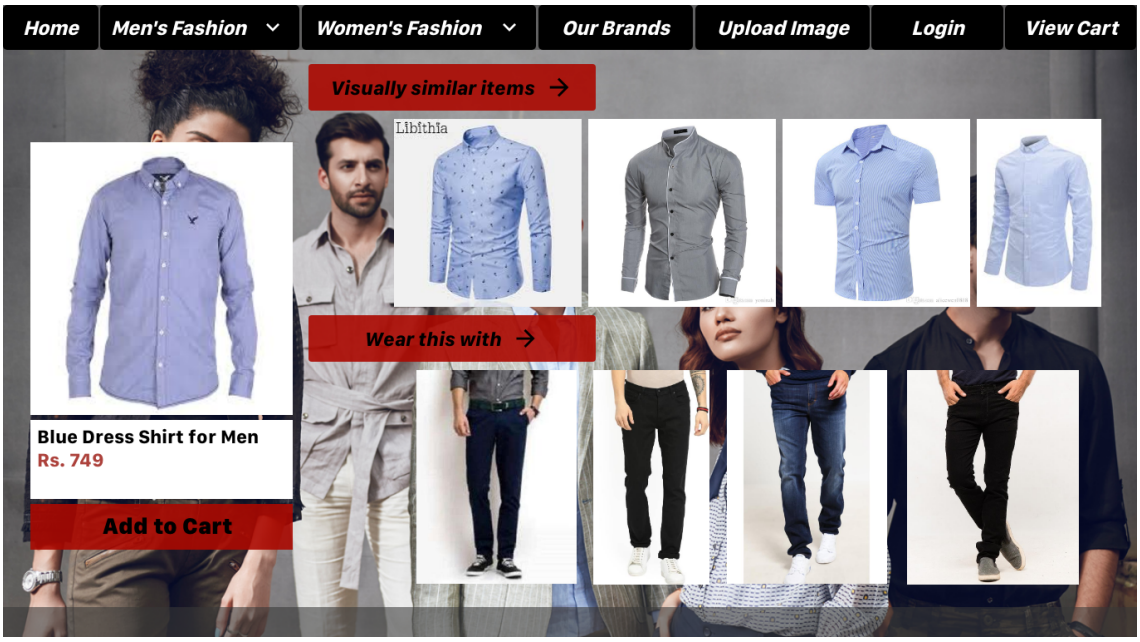

Figure 3.7: Recommendation Screen 1.0

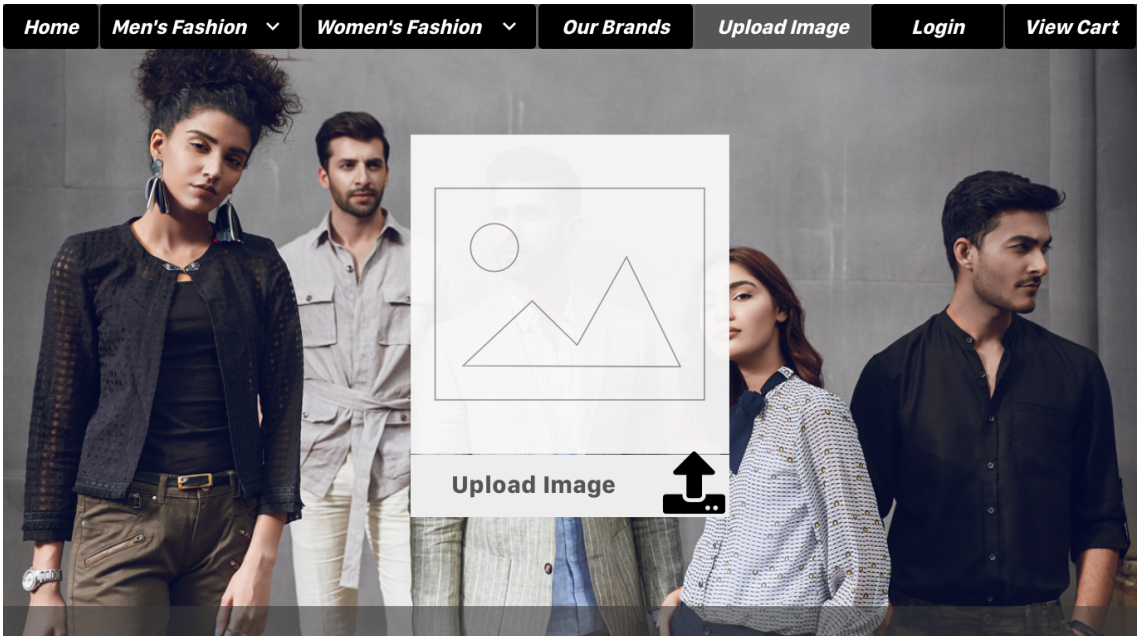

Figure 3.8: Upload Image Screen

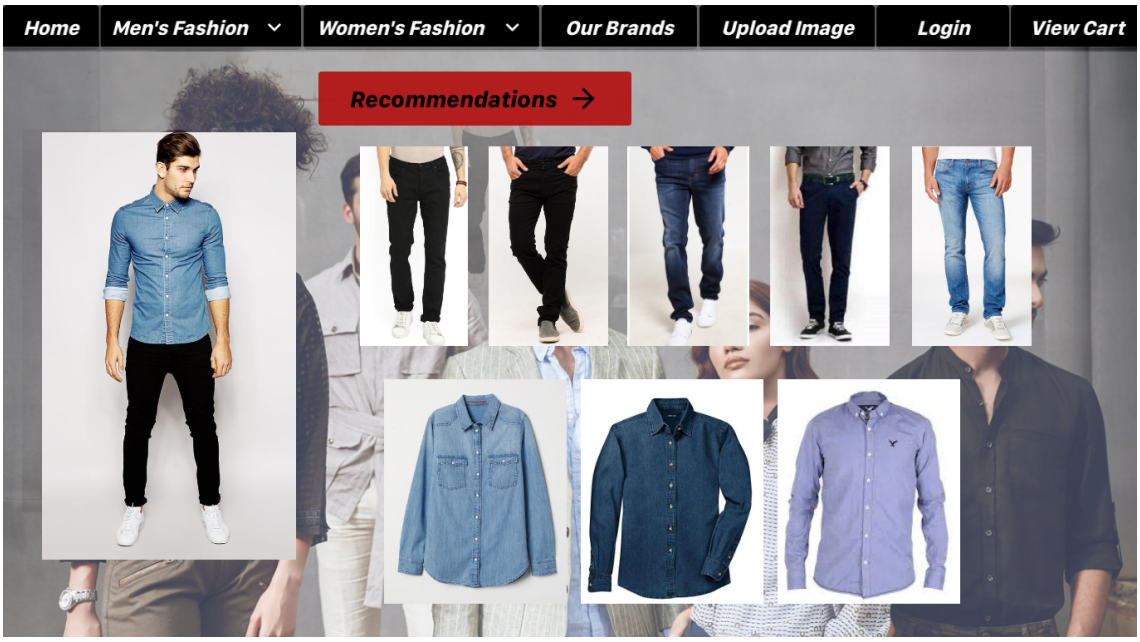

Figure 3.9: Recommendation Screen 2.0

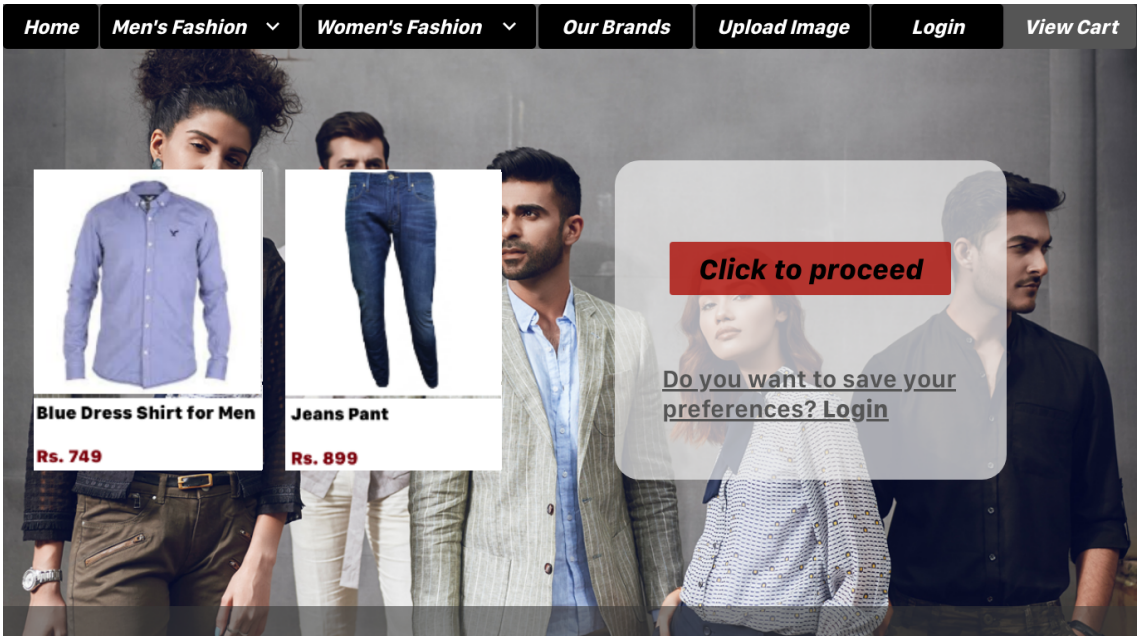

Figure 3.10: View Your Cart Screen

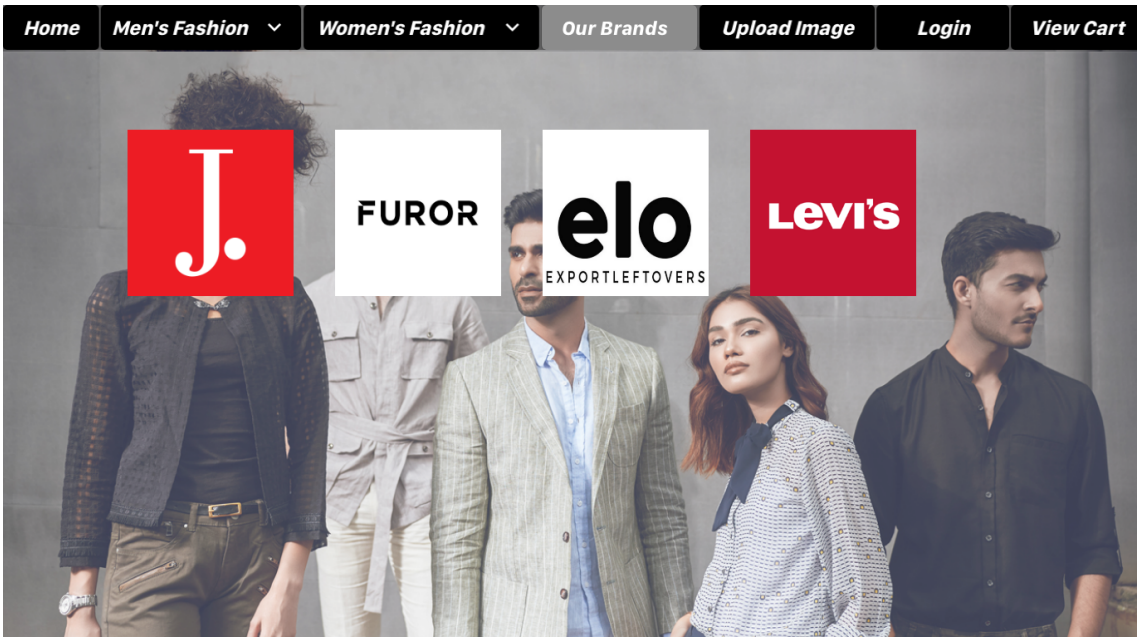

Figure 3.11: Partner Brands Screen

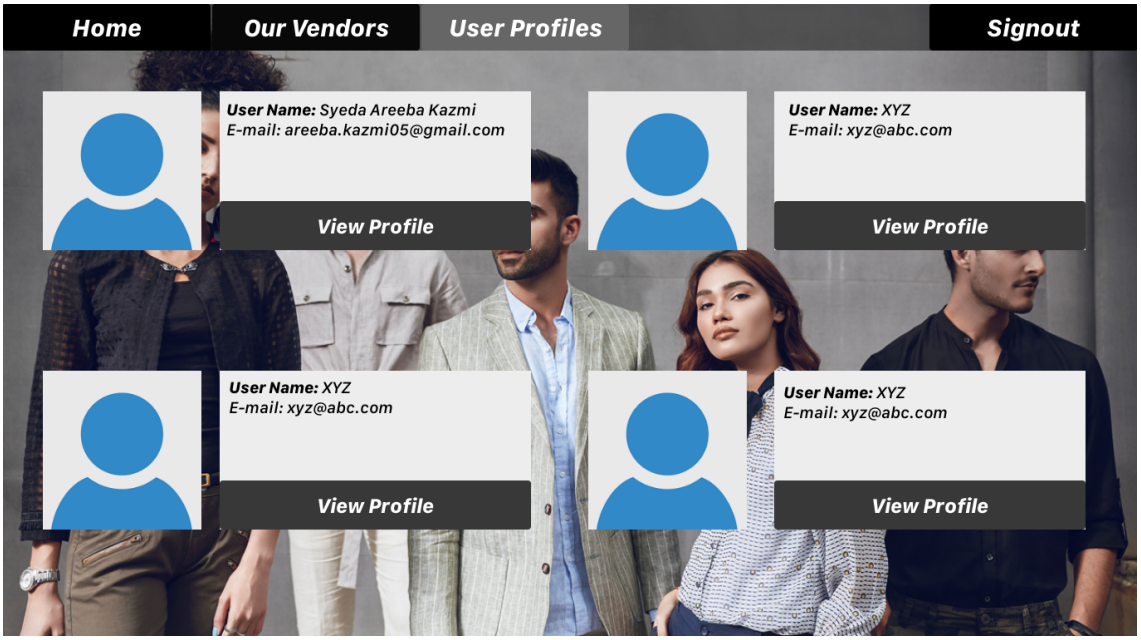

Figure 3.12: Admin-User Profiles Screen 1

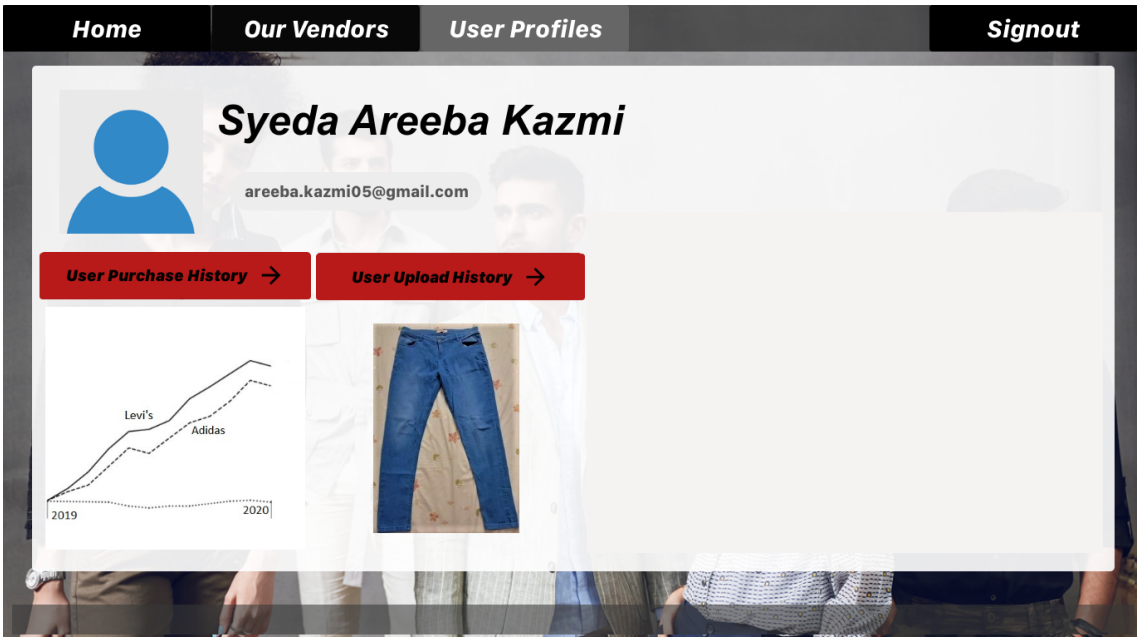

Figure 3.13: Admin-User Profiles Screen 1

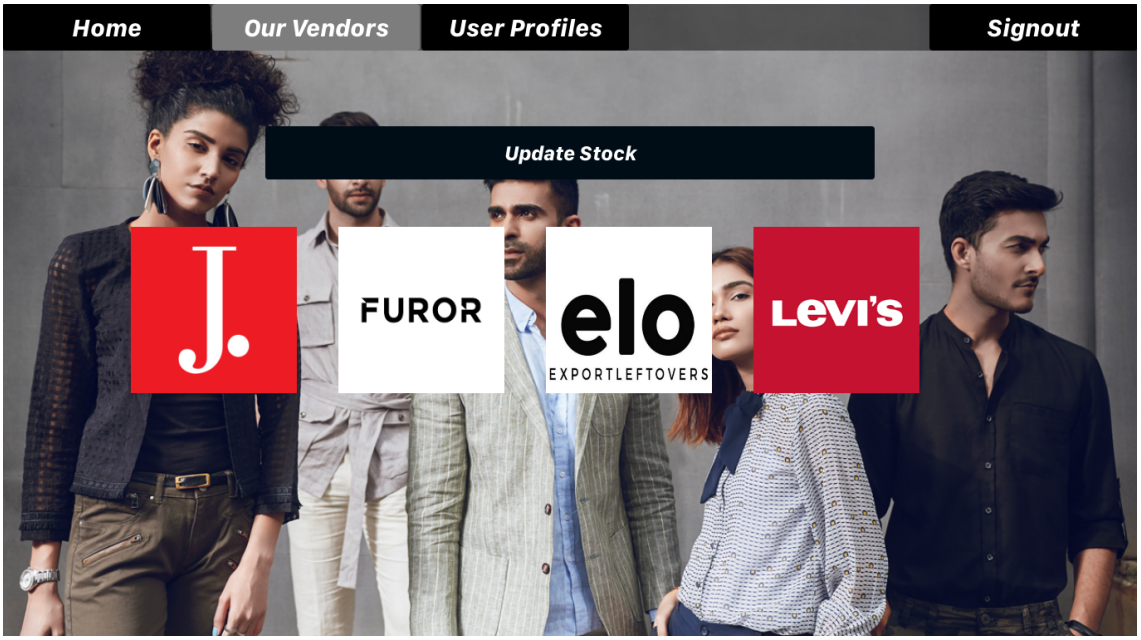

Figure 3.14: Admin Vendors Screen 1

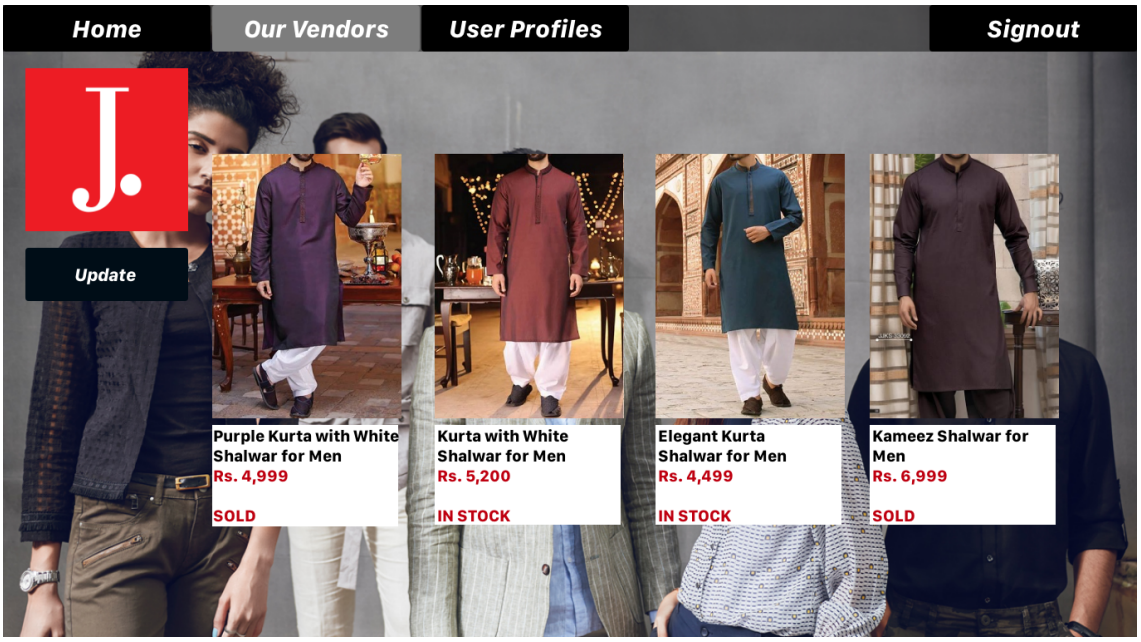

Figure 3.15: Admin Vendors Screen 2

3.3.2 Application Program Interface (API)

The system will be built using the Django framework at the server-end, and React.JS at the client-end (explained in detail in section 3.6). We aim to use REST (Representational State Transfer) APIs, served through the Django REST framework, for all client-server communication since RESTful services as an architectural style would lead to our application being lightweight and scalable. However, since graph-based communication channels are gaining more popularity and adaptation rates in internet technologies these days, and for reasons summarized in Table 2.3 ahead, our eventual goal is to establish an API based entirely off of GraphQL. So, for purposes of this web-application, our API will be written primarily as a RESTful service, but will be exposed to the client through a schema-first approach using a GraphQL wrapper. Figure 3.16 summarizes this.

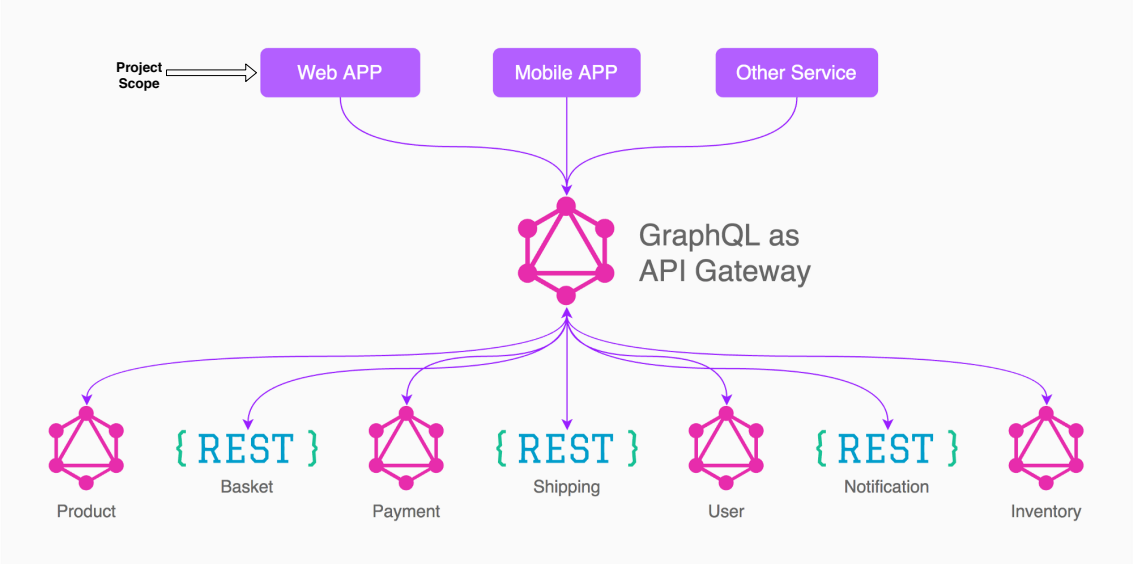

Figure 3.16: API Diagram

3.3.3 Communication Interfaces

Based on the project requirements and keeping the communication channels in mind, the chosen technology stack for the client-end is: TypeScript, React, and React Apollo (for interacting with the GraphQL API), and for the server-end is: Python, Django, and Graphene Django (for creating the GraphQL API). Table 2.1, Table 2.2,

and Table 2.3 offer a detailed comparison between why these communication interfaces were picked over some of their major competitors.

3.4 Use Cases

The two main users of the system are customers and the admin. Figure 3.17 summarizes their system use cases.

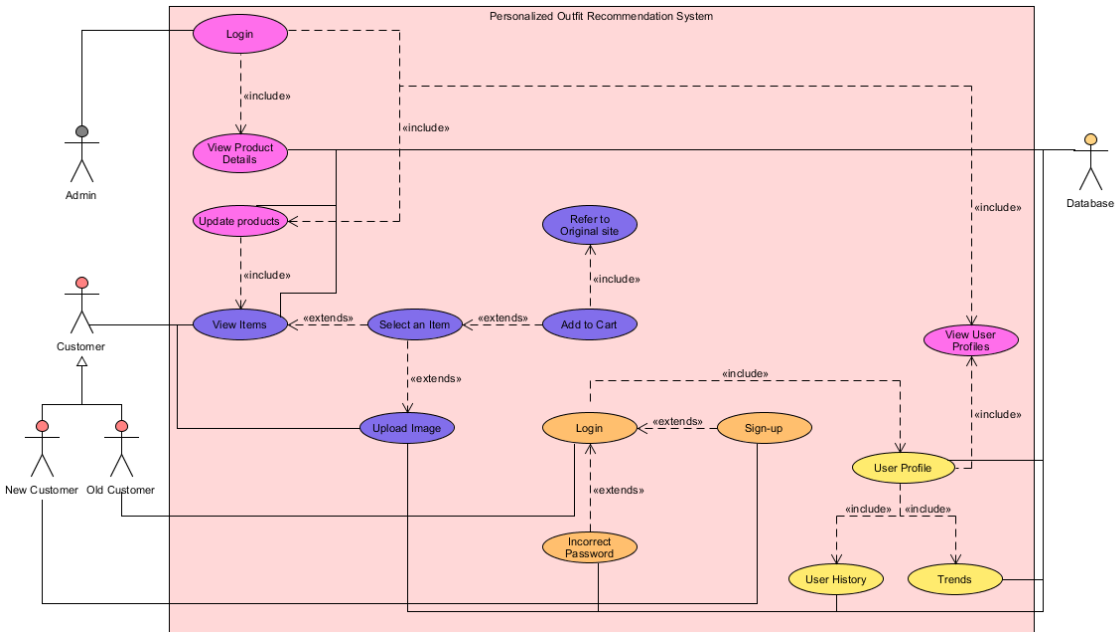

Figure 3.17: Use-Case Diagram

3.4.1 User Login

Use-Case Name:	Login
Use-Case ID:	PORS-01
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	Customer
Description:	System admin needs to login in order to view customer, product details and update the stock. Whereas, the customer needs to login to view the referred sites and save his profile
Pre-condition:	User has registered previously on the app and has valid email and password.
Trigger:	This use case is initiated when the admin wants to navigate the app or customer tries to view the referred site.
Typical course of events:	**Actor Action:** Actor enters valid username and password. **System Action:** Checks if the given username is registered in database and the given password is valid.
Alternate courses:	If the credentials are invalid then an error message is displayed.
Conclusion:	User successfully logs in using valid credentials.
Post-condition:	User login is logged in the database.

3.4.2 View Product Details

Use-Case Name:	View Product Details
Use-Case ID:	PORS-02
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	Customer
Description:	System Admin will view the product details to be able to update the stock. Whereas the customer will be able to see the product with details in order to select the items of their choice.
Pre-condition:	Logged in as Admin or Customer.
Trigger:	This use case is triggered when the user clicks a product.
Typical course of events:	Appropriate product details will be displayed to the user.
Alternate courses:	None
Conclusion:	The user is able to see the product detail page.
Post-condition:	User search is logged in the database.

3.4.3 Update Stock

Use-Case Name:	Update Stock
Use-Case ID:	PORS-03
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	None
Description:	System Admin will be able to update new stock and product details in case a product is sold out.
Pre-condition:	Logged in as Admin.
Trigger:	This use case will be triggered when the system admin clicks update stock button.
Typical course of events:	**System Action:** System will check all the products and update the sold out and new products accordingly.
Alternate courses:	None
Conclusion:	The stock is successfully updated.
Post-condition:	Customers are able to view the updated stock.

3.4.4 Upload an Image

Use-Case Name:	Upload an Image
Use-Case ID:	PORS-04
Priority:	High
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Application will allow customers to upload an image of their piece of clothing in order to find a similar or complementary items.
Pre-condition:	None
Trigger:	This use case will be triggered when the user will click upload image option.
Typical course of events:	**System Action:** System will check if the uploaded photo is of a piece of clothing
Alternate courses:	If the uploaded image is not recognizable, a suitable error message is displayed.
Conclusion:	Customer's valid picture will be uploaded on the app.
Post-condition:	Customer will be able to see the recommendation according to the uploaded picture.

3.4.5 Add to Cart

Use-Case Name:	Add to Cart
Use-Case ID:	PORS-05
Priority:	High
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Items selected by the customer will be added to the cart in order to proceed to the original referral site.
Pre-condition:	None
Trigger:	This use case will be triggered when the customer will select an item and click the option 'add to cart'.
Typical course of events:	System will check if the product is available to be added to the cart.
Alternate courses:	None
Conclusion:	Selected item will be added to the customer's cart.
Post-condition:	Valid referral link to the item is updated.

3.4.6 Refer to Original Sites

Use-Case Name:	Refer to Original Sites
Use-Case ID:	PORS-06
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Once the customer has selected items of his choice, those will be added to the cart and then he would be redirected to the original product site.
Pre-condition:	Items added to the cart
Trigger:	This use case will be triggered when a customer clicks on 'proceed to original site' option, available in his cart.
Typical course of events:	**System Action:** System will direct the customer to the original sites of the items available in customer's cart
Alternate courses:	None
Conclusion:	Customer directed to the original sites of the items he has chosen.
Post-condition:	Reference point is logged in the dataset.

3.4.7 Sign-up

Use-Case Name:	Sign-up
Use-Case ID:	PORS-07
Priority:	High
Primary Business Actor:	Admin
Other Participating Actors:	Customer
Description:	Admin and Customers will have to sign-up and register on the app in order to login and proceed.
Pre-condition:	Valid information by the user.
Trigger:	This use case will be triggered when a new user wants to register on the application.
Typical course of events:	**Actor Action:** Providing valid details. **System Action:** Checking if the details provided are available in the database.
Alternate courses:	If the details provided are incomplete or already registered then appropriate error message must be displayed.
Conclusion:	User is able to sign-up successfully.
Post-condition:	New user is added to the database.

3.4.8 Incorrect Password

Use-Case Name:	Incorrect Password
Use-Case ID:	PORS-08
Priority:	High
Primary Business Actor:	Admin, Customer
Other Participating Actors:	None
Description:	While logging-in on the app, the user must enter a valid email and password.
Pre-condition:	Registered on the app.
Trigger:	This use case will be triggered when a user tried to login with incorrect password or username.
Typical course of events:	**System Action:** System checks if the given password or user-name is registered in the database.
Alternate courses:	System displays an error message about incorrect credentials.
Conclusion:	User unable to login to the app.
Post-condition:	Incorrect login attempt is logged in the database.

3.4.9 User Profile

Use-Case Name:	User Profile
Use-Case ID:	PORS-09
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	System Admin
Description:	All registered users will have user profiles displaying their details, purchase history and uploaded image history.
Pre-condition:	User must be logged in.
Trigger:	This use case will be triggered when the admin wants to look at a particular user's profile or the customer himself clicks at his profile.
Typical course of events:	System will display the updated user profile to the user.
Alternate courses:	None
Conclusion:	User profile displayed.
Post-condition:	None

3.4.10 User History

Use-Case Name:	User History
Use-Case ID:	PORS-10
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	System Admin
Description:	All registered users will have their user history comprising of purchase history and uploaded image history.
Pre-condition:	1. User is registered in the database. 2. User has either purchased or uploaded an image before.
Trigger:	This use case will be triggered when a user purchases an item or uploads any image.
Typical course of events:	System will update the user history when the user will purchase or upload an item.
Alternate courses:	None
Conclusion:	User history is updated at every purchase or upload, given that the user is logged in.
Post-condition:	None

3.4.11 View Trends

Use-Case Name:	View Trends
Use-Case ID:	PORS-11
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Different statistics will be displayed indicating the sales or ratings of each brand in last 30 days.
Pre-condition:	users have either purchased or rated an item from any brand.
Trigger:	This use case will be triggered when the user clicks trend option.
Typical course of events:	System will update the trend at each purchase or rating by the user.
Alternate courses:	None
Conclusion:	User is able to see the most updated trends.
Post-condition:	Catalog matrix is updated in the database.

3.5 Datasets

This section describes the specific dataset which will be used to build our system. An appropriate snapshot of the dataset is also included in . Futher details, whenever needed, will be presented in the appendix.

For purposes of the image classification pipeline for our recommendation engine, we have selected DeepFashion [11] as our preliminary dataset. The dataset consists over 800,000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos. What makes this dataset suitable towards our system is the fact that it is annotated with rich information of clothing items. Each image in this dataset is labelled with 50 categories, 1,000 descriptive attributes, bounding box and clothing landmarks, and this would kick-start the training of our classifier. A category-wise subset of the dataset is depicted in Figure 3.18.

It is important to clarify here that the DeepFashion dataset consists of product images from western stores and outlets, so the classifier would not generalize

efficiently to products in the local eastern domain, which is a requirement of our system. Moreover, there are limitations in the amount of products and datasets we can scrape from eastern fashion stores such as J., Export Leftovers, Zellbury, Daraz, etc. If we were to use these as the only data for training our classifier, the model would indubitably suffer from underfitting. In order to maintain a good fit, we will use the concept of Transfer Learning to re-train the classifier trained on DeepFashion on our local scraped dataset from the aforementioned local fashion stores.

Figure 3.18: The DeepFashion Dataset

3.6 System Diagram

Figure 3.19 gives an overview of different modules of our system.

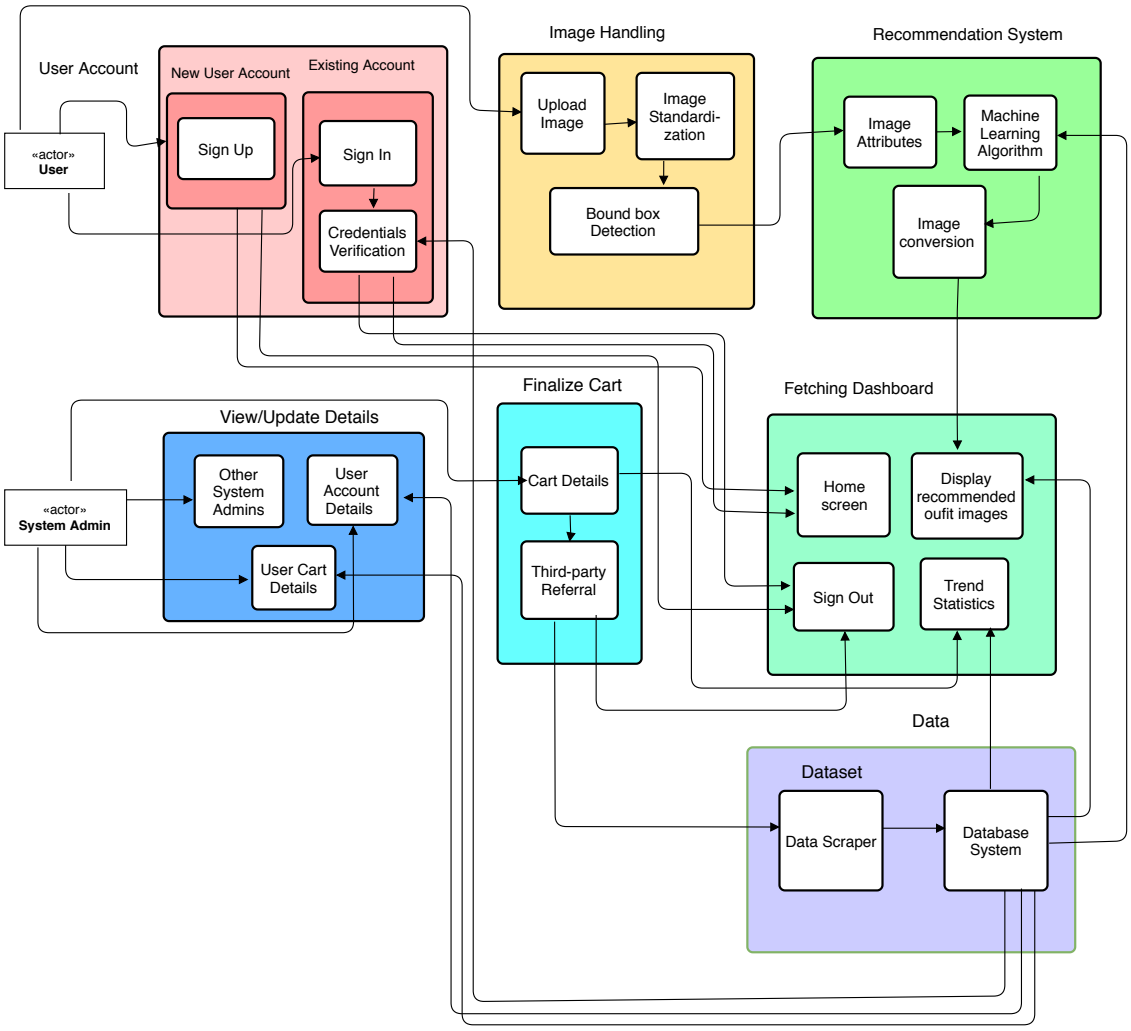

Figure 3.19: Module-wise System Diagram

Figure 3.20 gives an overview of the chosen architecture of our system.

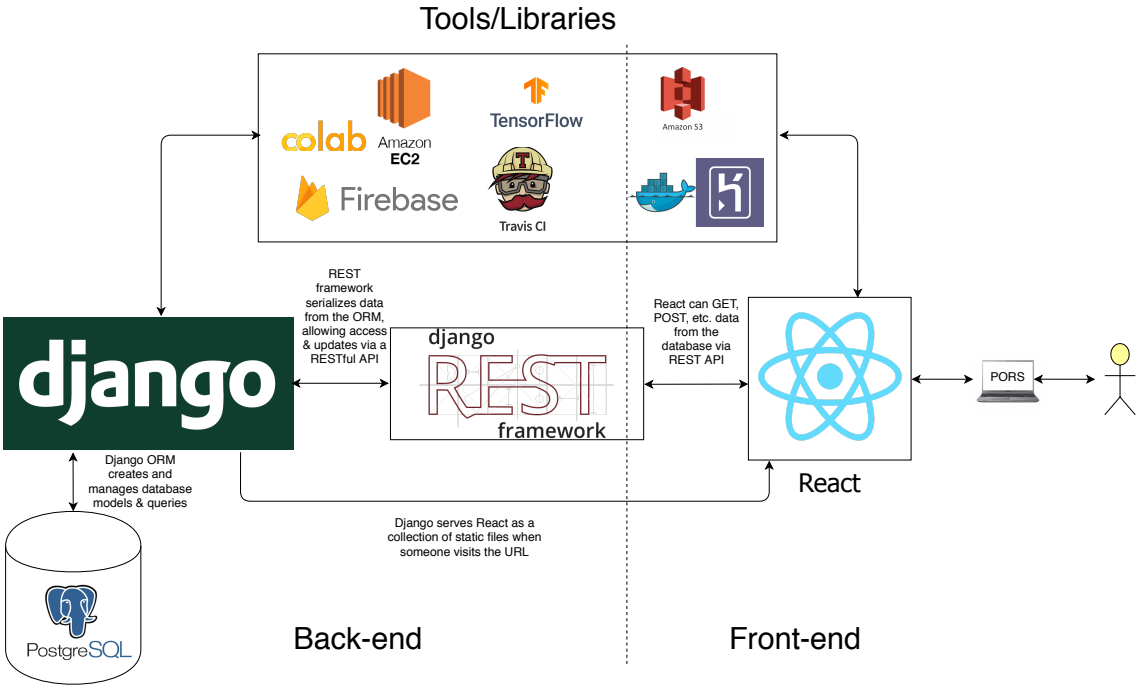

Figure 3.20: Technology-wise System Diagram

3.7 Data Flow Diagram

Rudimentary data flow diagrams for the system have also been constructed, given below in Figure 3.21 and Figure 3.22:

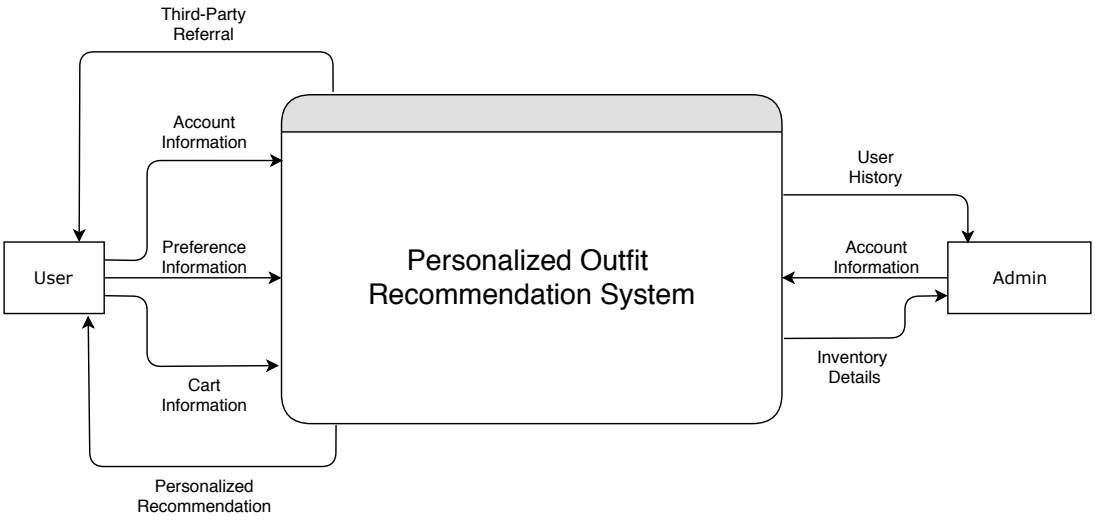

Figure 3.21: Context Level DFD

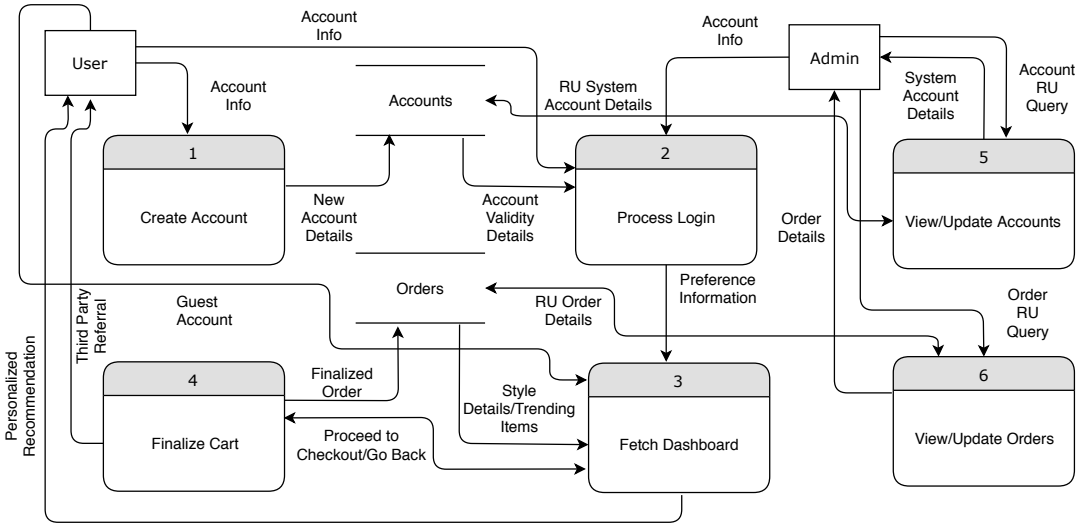

Figure 3.22: 0-Level DFD

3.8 Association Matrices

Association matrices for the system are depicted in Table 3.1, Table 3.2, and Table 3.3 ahead.

	User	System Admin
User	Individual	All
User ID	R	CRUD
Name	RU	CRUD
Email ID	CRU	RD
Password	RU	RD
Sign in/Sign up/ Sign out	Individual	All
Account ID	R	RD
Date	R	RD
Verify Credentials	Individual	All
Account Valid	R	CRUD
Logged In	R	CRUD
Date	R	CRUD
Fetch Dashboard	Individual	All
Page ID	R	CRUD
Image Details	R	CRUD
Finalize Cart	Individual	All
Cart ID	R	CRUD
Date	R	CRUD
Item No.	RU	CRUD
Third-party Referral	R	CRUD
Recommendation Details	Individual	All
Recommendation ID	R	CRUD
User ID	R	CRUD
Date	R	CRUD
Image	Individual	All
Image ID	R	CRUD
Category	R	CRUD
Image Attribute	R	CRUD
Recommendation ID	R	CRUD

Table 3.1: CRUD: Data to Location Matrix

	Account	Credentials	Dashboard	Cart	Image
User					5
User ID	R	R	R	R	R
Name	CRU	RU			
Email	CRU	R			R
Password	CRU	R			
Sign in/up/out					
Account ID	R	R			
Date	R	R			
Verify Credentials					
Account Valid		RU	R		
Logged In		RU	R		
Date		R	R		
Fetch Dashboard					
Page ID			R	R	
Image Details			RUD	RU	
Finalize Cart					
Cart ID			R	R	
Date			R	RUD	
Item No.			R	RUD	
Third-party Referral			R	RUD	
Recommendation Details					
Recommendation ID			R	RU	
User ID			R	RUD	
Date			R	RUD	
Image					
Image ID			R	R	R
Category			RUD	RUD	R
Image Attribute			RUD		R
Recommendation ID			R	R	R

Table 3.2: CRUD: Data to Process Matrix

	User	System Admin
Process Account	*	
Process Verify Credentials		*
Process Dashboard		*
Process Cart	*	
Process Image Handling		*

Table 3.3: CRUD: Process to Location Matrix

4. Software Design Specification (SDS)

This chapter provides important artifacts related to design of our project.

4.1 Data Design

This section presents the structure of our database that caters to persistent data storage in our project. The structure is shown as a normalized data model for relational databases. It clearly shows entities, attributes, relationships with their cardinalities, and primary and foreign keys. We have used DB designer to build our data model.

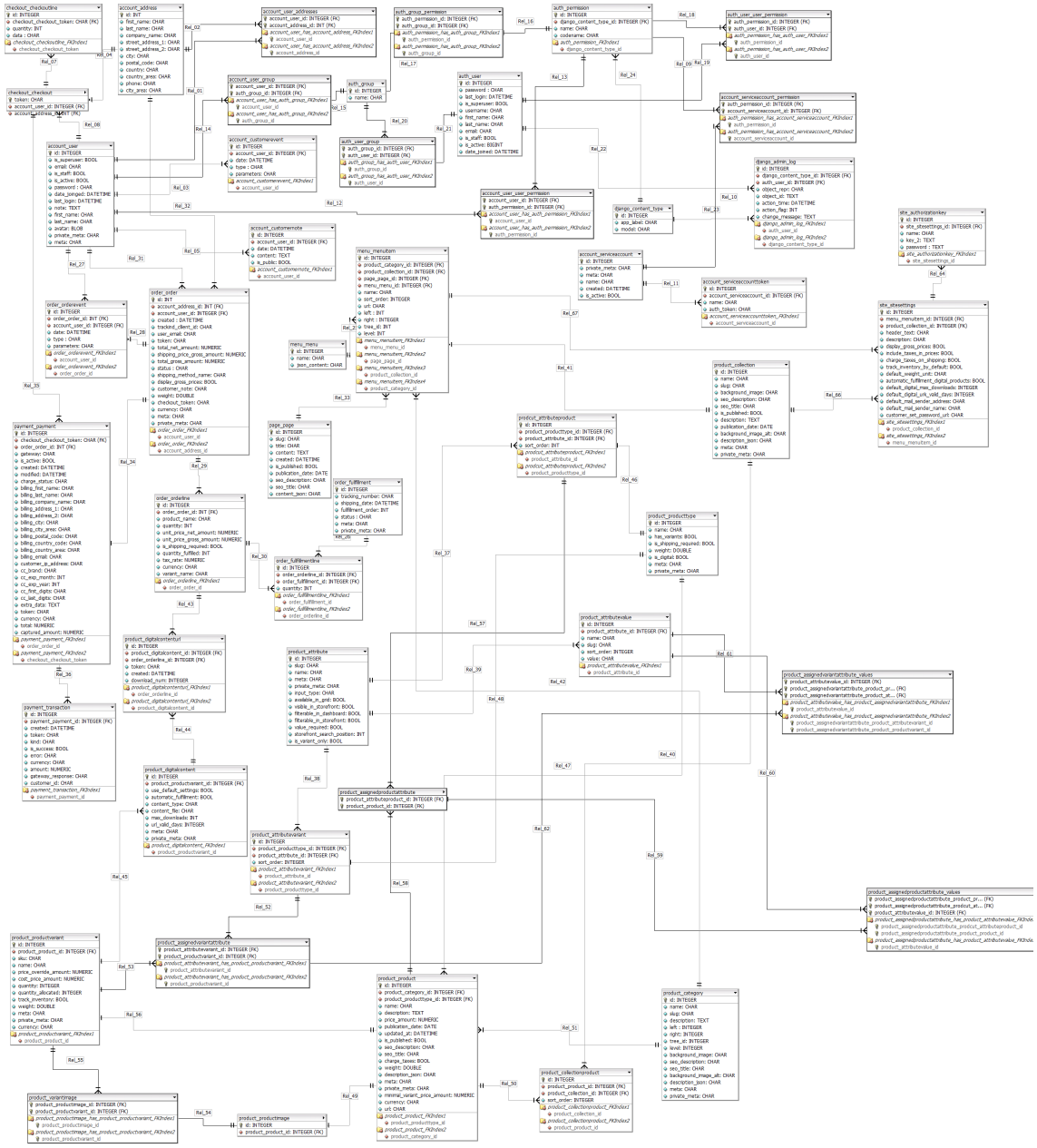

Figure 4.1: Entity Relationship Diagram

4.2 State Diagram

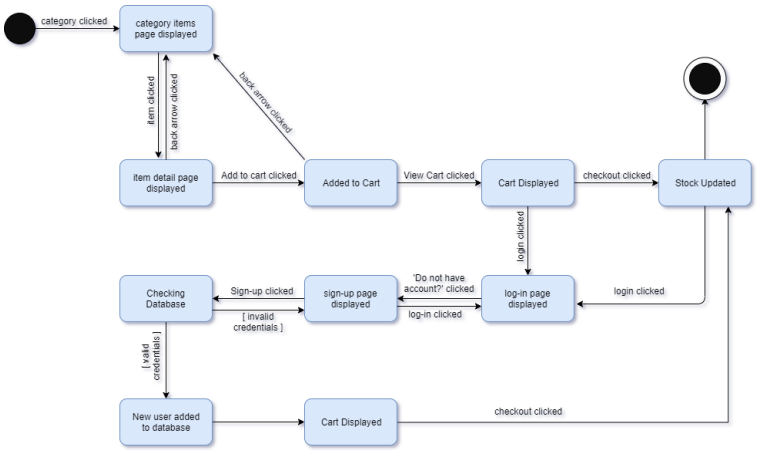

Figure 4.2: State Diagram 1

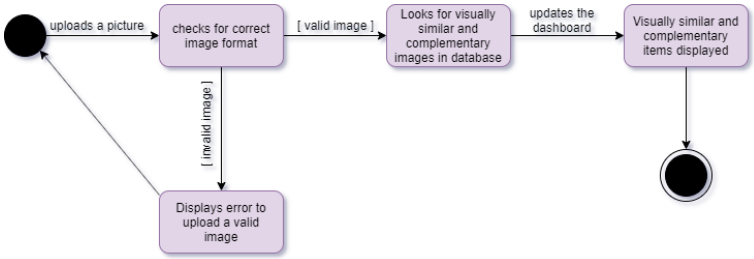

Figure 4.3: State Diagram 2

4.3 Sequence Diagram

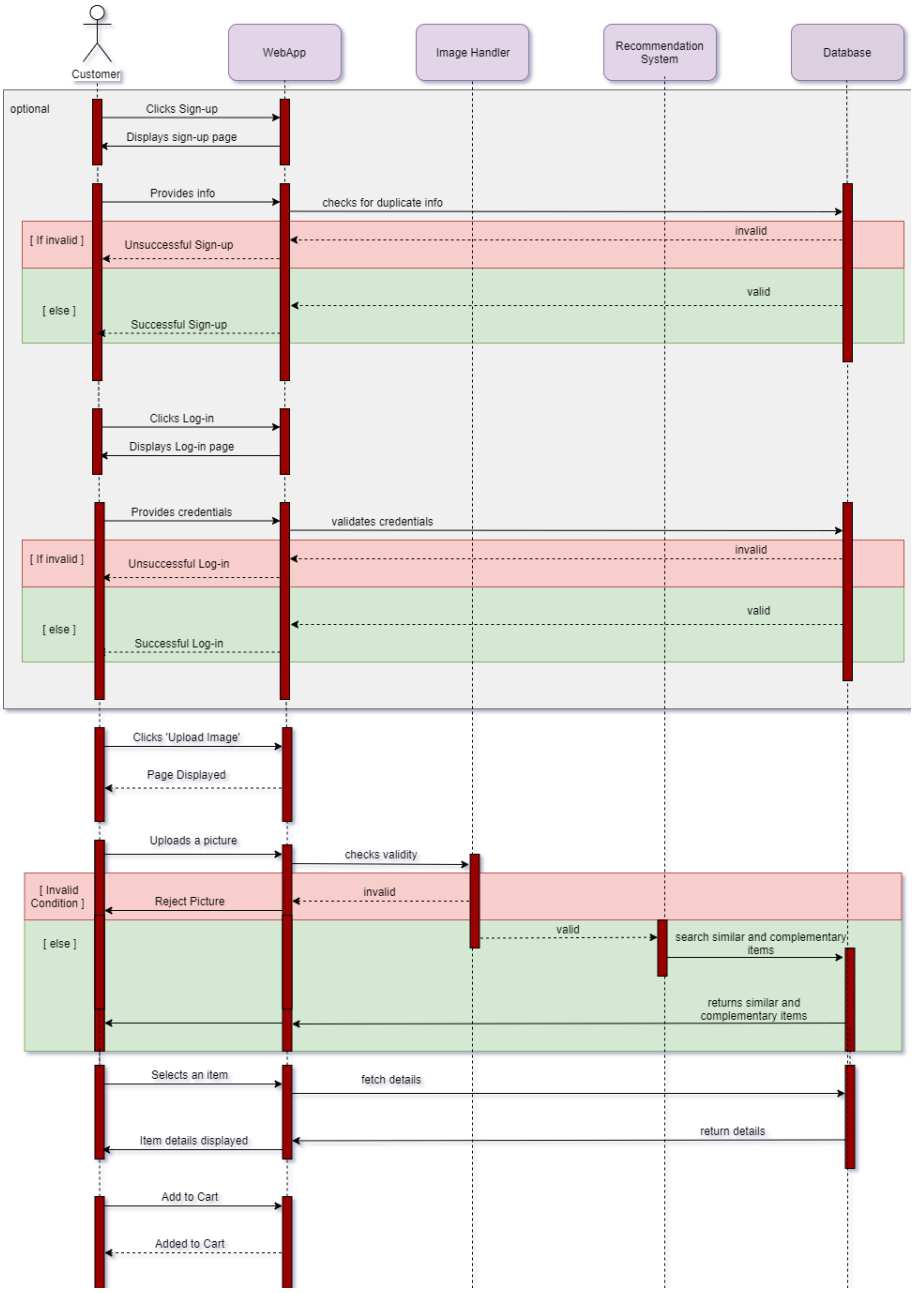

Figure 4.4: Sequence Diagram - Customer

5. Experiments and Results

TBA.

6. Conclusion and Future Work

TBA.

Appendix A. More Math

Here, we describe the background math for the techniques used in the text.

Appendix B. Data

Here is a dump of our 2TB data set. Enjoy!

Appendix C. Code

Here is our code. Bits over trees, courtesy of HEC!

```
print('Hello World!')
print('Computing true random number.')
print('Capturing interstellar radiation.')
print('This will take time!')
import random
import time
time.sleep(3600*random.randint(1,10))
print(4)
```

Our code can be found at this GitHub link.

References

[1] Salesforce. *The Power of Personalized Shopping*. URL: https://www.salesforce.com/form/commerce/power-of-personalized-shopping.

[2] McKinsey. *How Retailers Can Keep Up with Consumers*. URL: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

[3] The News Pakistan. *Pakistani Fashion Industry has Great Potential*. URL: https://www.thenews.com.pk/print/450217-pakistani-fashion-industry-has-great-potential.

[4] Digital Commerce 360. *US millennials now do most of their shopping online*. URL: https://www.digitalcommerce360.com/2019/03/26/millennials-online-shopping/.

[5] eMarketer. *Millennials Make More Apparel Purchases than Other Generations*. URL: https://www.emarketer.com/content/millennials-make-more-apparel-purchases-than-other-generations.

[6] A. Veit et al. *Learning visual clothing style with heterogeneous dyadic co-occurrences*. URL: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Veit_Learning_Visual_Clothing_ICCV_2015_paper.pdf.

[7] J. McAuley et al. *Image-based recommendations on styles and substitutes*. URL: https://arxiv.org/pdf/1506.04757.pdf.

[8] X. Yi Y. Hu and L. S. Davis. *Collaborative fashion recommendation: A functional tensor factorization approach*. URL: http://delivery.acm.org/10.1145/2810000/2806239/p129-hu.pdf.

[9] Xufeng Han et al. *MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Han_MatchNet_Unifying_Feature_2015_CVPR_paper.pdf.

[10] Ziwei Liu et al. *DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Liu_DeepFashion_Powering_Robust_CVPR_2016_paper.pdf.

[11] Ziwei Liu et al. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.